RISC-V Big Endian

RISC-V EU Summit 2025

CODETHINK

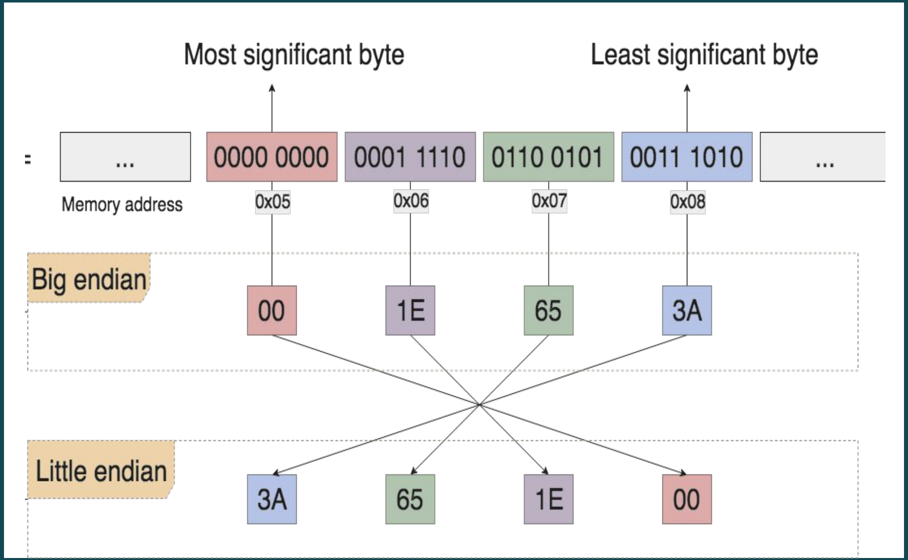
01
Introduction

Endianness

What is Endianness?

- The order values are composed in memory
- Big Endian (BE): **left-to-right**
- Little Endian (LE): **right-to-left**

An easy way to remember: Big/Little endian stores the Big/Little end of the number in the first memory address.



Different Endians? Different Users?

Big Endian:

- Often used in older architectures and applications
- Many networking protocols use BE. LE systems have to reverse byte order (adding overhead)

Little Endian:

- x86 chose little endian as at the time, it made operations such as type casting is easier (as memory layout doesn’t change)
- x86 used it and achieved dominance

Why are we interested?

Current state:

- RISC-V is LE
- Latest ISA specification adds runtime configurable endianness
  - Part of ISA volume II, privileged architecture
  - Configuration added (version 1.12, 2022)

Definition of Done:

- Overall:
  Linux running on Qemu in big endian
- Qemu → Add ability to configure CSRs and modify data accesses
- OpenSBI → Boot system in big endian and correctly deal with IO
- Linux → Build minimal kernel and userland

02
QEMU

CPU Flags

- Goal is to update the CSRs that controls the endian for each execution level (M/S/U)
  - M-mode is controlled by bit 37 (MBE) of the `mstatus` CSR
  - MBE is then read-only cloned into bit 36 (SBE) and bit 6 (UBE)
- CSR writes allowed to `mstatus` bits
  - Writes update new context field: `ctx->be_data`
- For QEMU: `sstatus` SBE and UBE are read-only clones of `mstatus`
  - We may add SBE/UBE configuration later

```
#define MSTATUS_MBE 0x2000000000ULL
```

Load & Store

- Add memory operation flag for load & store operations

```
if (ctx->be_data) { mop |= MO_BE; }
```

- Same for *Zacas* extension (Atomic Compare-and-Swap instructions)
- FPU follows integer endian
- MMU translation follows S mode endian
- Some wrangling of byte ordering

```
cpu_to_le64(), cpu_to_be64()
be64_to_cpu(), le64_to_cpu()
```

03

Software stack

Software updates

- MMIO bus is still little endian → must swap IO data
- Instructions are always little endian
  - Hand assembly of instructions can be problematic
  - Introspection and runtime modification needs changing
- Updated buildroot selection menu to add big endian option
- Software such as uclibc needed build updates due to assumed little endian

OpenSBI

- Updated build to support big endian
- Added an option and corresponding code to change harts to big endian
  - *Switch M-mode to big endian at start by setting* `mstatus.MBE` *bit early in the initialisation code and when hart is re-initialised*
  - *Added an option to set* `mstatus.{U,S}BE` *flags when the hart is initialised for big endian S and U modes*
- Updated IO code for endian swap

OpenSBI: config and CSR set patch

```
diff --git a/Kconfig b/Kconfig
        help
          Say Y here if you like fun challenges

+config OPENSBI_BE_SET
+       bool "Set M-Mode to be big-endian in startup code"
+       depends on OPENSBI_BE

diff --git a/firmware/fw_base.S b/firmware/fw_base.S
 _start:
+#ifdef CONFIG_OPENSBI_BE_SET
+#if __riscv_xlen == 64
+       li      s0, MSTATUS_MBE
+       csrs    CSR_MSTATUS, s0
+#endif
+#endif
diff --git a/lib/sbi/sbi_hart.c b/lib/sbi/sbi_hart.c
@@ -39,6 +39,10 @@ static void mstatus_init(struct sbi_scratch *scratch)

+#ifdef CONFIG_OPENSBI_BE_SET
+       mstatus_val |= MSTATUS_MBE;
+#endif
```

Some parts omitted for space/clarity

Instruction endian

The instruction stream is always little endian

OpenSBI uses `.word` for creating instructions the assembler may not support. This stores the instruction in data endian. This is fixed by changing `.word` to `.insn` to mark these as instructions and to not be endian swapped.

This has the bonus of instructions being marked as instructions!

```
diff --git a/lib/sbi/sbi_hfence.S b/lib/sbi/sbi_hfence.S
@@ -32,7 +32,7 @@ __sbi_hfence_gvma_vmid_gpa:
         * HFENCE.GVMA a0, a1
         * 0110001 01011 01010 000 00000 1110011
         */
-       .word 0x62b50073
+       .insn 0x62b50073
        ret
```

Build time endian checks

Packages like ucLibc assume little endian RISC-V. This is fixed by using the pre-defined macro **__BYTE_ORDER__** which defines the endian

```
diff --git a/libc/sysdeps/linux/riscv64/bits/endian.h

+/* extract byte order from gcc predefines */
+#if defined(__BYTE_ORDER__)
+#if __BYTE_ORDER__ == __ORDER_BIG_ENDIAN__
+#define __BYTE_ORDER __BIG_ENDIAN
+#endif
+#if __BYTE_ORDER__ == __ORDER_LITTLE_ENDIAN__
+#define __BYTE_ORDER __LITTLE_ENDIAN
+#endif
+#endif
```

Data decomposition

- Often used in optimised or acceleration code
  - Modify shifts and masks when accessing fields
  - Change struct definitions

For example, in big endian the first byte of the string is at the top of the data

```
struct tcphdr {
       ...
# if __BYTE_ORDER == __LITTLE_ENDIAN
        uint8_t th_x2:4;        /* (unused) */
        uint8_t th_off:4;       /* data offset */
# endif
# if __BYTE_ORDER == __BIG_ENDIAN
        uint8_t th_off:4;       /* data offset */
        uint8_t th_x2:4;        /* (unused) */
# endif
```

Linux Kernel

- Some assembly code changed to use simplified/non-accelerated versions
- Fixed instruction modification/introspection code
  - Static branches
  - BUG() macro uses illegal instruction trap
  - BPF's BSWAP code needed update
  - BPF's JIT correctly emit little endian instructions
  - {K,U}probe code modification updates

04
Future

Problems

1. The `riscv64be-uclibc-gcc` will not build libnptl (segfaults gcc)
2. The buildroot/musl has issues possibly with library code
3. The buildroot/glibc has not been tested
4. Kernel kprobes not working

Upstreaming

1. Sent RFC to lists: kernel, `qemu-riscv`
2. Someone already sent `zbb rev8` swap patch for Linux kernel
3. Conference talks to spread interest
4. Please get involved

For more information, project source, patch submissions, and logs can be found on our GitLab. Scan the QR code to access.

gitlab.com/CodethinkLabs/riscv_bigendian

Thank You.

Codethink Ltd.

3rd Floor Dale House,
35 Dale Street,
MANCHESTER,
M1 2HF,
United Kingdom